Far East Journal of Experimental and Theoretical Artificial Intelligence
Volume 2, Issue 2, 2008, Pages 129-140

PARTICLE SWARM OPTIMIZATION FOR DISTRIBUTED SYSTEM RELIABILITY

CHIN-CHING CHIU

Department of Management Information System
Takming University of Science and Technology
Taipei, Taiwan, R.O.C.
e-mail: chiu@takming.edu.tw

Abstract

The reliability optimization of a distributed system is generally an NP-hard problem. The reliability of a k-node set with capacity constraint is defined as the probability that a subset of the set of processing elements in system is connected and possesses sufficient node capacity. In this work, a particle swarm optimization is presented to reduce the computational time and the absolute error from the exact solution for obtaining the reliability of a k-node set with capacity constraint. The method uses an efficient objective function to evaluate the chosen nodes when a particle derives its k-node set process. Reliability computation is performed only once, thereby spending less time to compute the reliability. Moreover, the absolute error of proposed algorithm from exact solution is smaller than that of k-tree reduction method. Computational results demonstrate that the proposed algorithm is a more efficient solution for a large distributed system than conventional ones.

1. Introduction

Particle swarm optimization (PSO) was first introduced by Kennedy and Eberhart as an optimization technique for continuous space problems in 1995 [4]. It is a kind of evolutionary computation technique motivated

---

Keywords and phrases: particle swarm optimization, distributed systems, reliability optimization.

Communicated by Zne-Jung Lee

Received November 25, 2008

by the behavior of organisms such as bird flocking. Because of its simple and easy implementation, PSO has been widely applied to optimization problems which have been tackled by other evolutionary computation techniques such as genetic algorithms and evolutionary programming.

The economic benefits of resource sharing primarily enlarge for the importance of distributed system (DS). A DS focuses on providing efficient communication among various nodes, thereby increasing their reliability and making their service available to more users [12]. Designing such systems must consider system reliability, which heavily relies on the topological layout of the communication links [10]. The network reliability problem with respect to a network with a general structure is NP-hard [13]. Efficient algorithms easily implemented on a computer are needed to analyze the reliability of large networks. In addition, such algorithms should yield good appoximations of the reliability when the networks are so large that the computational time becomes prohibitive.

This work largely focuses on how to compute nearly maximum system reliability subject to the capacity constraint. In the k-tree reduction method (KM) [1], the starting node is the first node v_1. To select other adequate nodes in a sequential manner depends on the maximum product of reliability by capacity of the k-node set with another node until the capacity constraint is satisfied. The number of reliability computation is still large. In addition, the above product heavily relies on the total capacity of each node but only slightly depends on the k-node set reliability; therefore, it barely derives the optimal solution.

This paper presents a particle swarm optimization to obtain an adequate k-node set. Then SYREL [3] is applied to compute the reliability. For a large DS on various DS topologies, our results demonstrate that the proposed algorithm is more reliable and efficient than conventional algorithms, the exact method (EM) [2] and the KM in terms of execution time.

2. Problem Description

Bi-directional communication channels operate between processing elements. A DS can be modeled by a simple undirected graph $G = (V, E)$, where V denotes a set of processing elements and E represents a set of communication links.

A k-node set reliability problem can be characterized as follows:

Given

Topology of a DS.

The reliability of each communication link.

The capacity of each node.

A set of data files.

Assumption

Each node is perfectly reliable.

Each link is either in the working (ON) state or failed (OFF) state.

Constraint

The total capacity of data files to be allocated.

Goal

To select a specified set K of nodes in a DS to allocate data files, by doing so,

k-node set reliability is adequate under constraints.

A set, K, of nodes can be derived from the given vertices set V (the total number of vertices $|V| = n$) that constitutes a DS in that k-node set G_k reliability $R(G_k)$ is adequate and the total capacity satisfies the capacity constraint $C_{\min}$. The main problem can be mathematically stated as follows:

Object: Maximize $R(G_k)$

subject to $\sum_{v_s \in G_k} c(v_s) \geq C_{\min}$

where $c(v_s)$ is the capacity of the sth node.

Obviously, the problem for a large DS, as in a metropolitan area network, requires a large execution time.

3. PSO for *k*-node Set Reliability

In this section, we present a PSO algorithm to maximize system reliability.

3.1. The concept of proposed algorithm

The EM, an optimal solution, requires excessive execution time in a large DS and cannot effectively reduce the problem space. Occasionally, an application requires an efficient algorithm to compute the reliability due to its resource considerations. In these circumstances, achieving optimal reliability may not be desired. Instead, an effective algorithm with an approximate reliability is highly attractive. In fact, most DS are large and an increasing number of nodes causes exponential growth of the execution time for a solution. Although able to reduce computational time, the KM has much difficulty in deriving the optimal solution. Therefore, this work presents a PSO algorithm to reduce the total execution time to achieve the optimal *k*-node set reliability of DS.

The reliability of a set of selected nodes depends on their links and the link reliability. For any node, the degree $d(\mathrm{v}_s)$ of that node v_s affects the number of paths of the information can be transferred from others' nodes. The following formula is used to compute the weight of node v_s where $q_{s,d}$ denotes the probability of failure of link $e_{s,d}$.

$$w(\mathrm{v}_s) = 1 - \prod_{z=1}^{d(\mathrm{v}_s)} q_{s,k_z}. \tag{1}$$

In the network, two nodes may contain many paths between them. A path's length is between one and $n-1$. To reduce the computational time, we consider the path in which the length is not greater than two. The following formula is used to evaluate the weight of link $e_{s,d}$ which the probability of success is $p_{s,d}$. The $y_{s,d}$ denotes the number in which the length of a path between v_s and v_d is two.

$$w(e_{s,d}) = 1 - q_{s,d} \prod_{z=1}^{y_{s,d}} (1 - (p_{s,k_z} p_{k_z,d})). \tag{2}$$

In the same manner, if no direct link exists between ν_s and ν_d, but there are at least two paths whose length is two between them ($\varphi_{s,d}$ denotes the number of those paths), the following formula is used to evaluate the weight of $\varepsilon_{s,d}$.

$$w(\varepsilon_{s,d}) = 1 - \prod_{z=1}^{\varphi_{s,d}} (1 - (p_{s,k_z} p_{k_z,d})). \tag{3}$$

In each set of nodes, if the number of members of a set is $|k|$, the following formula can be used to compute its weight value.

$$w(G_k) = \left\{ \left[\sum_{e_{s,d} \in G_k} w(e_{s,d}) + \sum_{\substack{e_{i,j} \notin G_k, \\ \nu_s, \nu_d \in G_k}} w(\varepsilon_{s,d}) \right] / [|k|(|k|-1)/2] \right.$$

$$\left. + \sum_{\nu_s \in G_k} w(\nu_s)/[(n-1) \times |k|] \right\} / \sqrt{|k| + 2}. \tag{4}$$

The penalty function of selected set is defined as follows.

$$\nabla(G_k) = \begin{cases} 0 & \text{if } \sum\limits_{\nu_s \in G_k} c(\nu_s) \geq C_{\min} \\ C_{\min} - \sum\limits_{\nu_s \in G_k} c(\nu_s) & \text{if } \sum\limits_{\nu_s \in G_k} c(\nu_s) < C_{\min}. \end{cases} \tag{5}$$

In a PSO system, a swarm of individuals (called *particles*) fly through the search space. Mendes et al. [8] discusses the complete information about the particle swarm optimization. In the standard PSO algorithm, all particles have their position, velocity, and fitness values. Let *ps* denote the swarm population size, $X^t = [X_1^t, X_2^t, ..., X_{ps}^t]$ represents population during t iteration. Then each particle in the swarm population has a current position $X_i^t = [x_{i,1}^t, x_{i,2}^t, ..., x_{i,n}^t]$ in the n-dimension space as a candidate solution, a current velocity $\Psi_i^t = [\psi_{i,1}^t, \psi_{i,2}^t, ..., \psi_{i,n}^t]$ which a constant $\Psi_{\max}$ (often set to 4) is used to limit the range of $\psi_{i,j}^t$, i.e., $\psi_{i,j}^t \in [-\Psi_{\max}, \Psi_{\max}]$ for avoiding the particle to converge to local optima, a particle best position $pbest_i^t = [\rho_{i,1}^t, \rho_{i,2}^t, ..., \rho_{i,n}^t]$, and a global best position $gbest^t = [g_1^t, g_2^t, ..., g_n^t]$ of the swarm population until current iteration.

The current velocity of the jth dimension of the ith particle is updated as follows:

$$\psi_{i,j}^t = \mu^{t-1}\psi_{i,j}^{t-1} + c_1 r_1(\rho_{i,j}^{t-1} - x_{i,j}^{t-1}) + c_2 r_2(g_j^{t-1} - x_{i,j}^{t-1}), \tag{6}$$

where c_1 and c_2 are acceleration coefficients which were often set to be 2.0 according to past experience and r_1 and r_2 are uniform random numbers between $[0, 1]$. Thus, the particle flies through potential solutions toward $pbest_i^t$ and $gbest^t$ in a navigated way while still exploring new areas by the stochastic mechanism to escape from local optimal. Since c_1 expresses how much the particle trusts its own past experience, it is called the *cognitive parameter*, and since c_2 expresses how much it trusts the swarm, it is called the *social parameter*. The inertia weight μ^{t-1} which is a parameter to control the impact of the previous velocities on the current velocity. The inertial weight can be dynamically varied by applying an annealing scheme for the μ-setting of the PSO, where μ decreases from $\mu = 0.9$ to $\mu = 0.4$ over the whole run. In general the inertia weight μ^t is set according to the following equation [11].

$$\mu^t = \mu_{\max} - \frac{\mu_{\max} - \mu_{\min}}{\text{iter}_{\max}} \times (t - 1), \tag{7}$$

where $\mu_{\max}$ and $\mu_{\min}$ are both numbers called *initial weight* and *final* *weight* respectively. In addition, $\text{iter}_{\max}$ is the maximum number of iterations, and t is the current iteration.

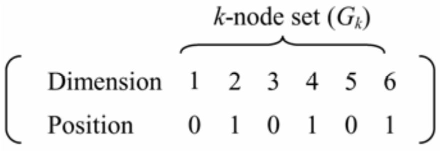
In the above discussion, PSO is restricted in real number space. However, many optimization problems are set in a space featuring discrete or qualitative distinctions between variables. Discrete PSO essentially differs from the original PSO in two characteristics. First, the particle is composed of the binary variable. Second, the velocity must be transformed into the change of probability, which is the chance of the binary variable taking the value. Thus, in the discrete binary version [5-7, 9], a particle moves in a state space restricted to zero and one on each dimension, where each $\psi_{i,j}$ represents the probability of bit $x_{i,j}$ taking the value 1. Thus, the step for updating $\psi_{i,j}$ remains unchanged as shown in Eq. (6), except that $\rho_{i,j}$ and g_j are integers in $\{0, 1\}$ in binary case. The resulted changes in position are defined as follows:

$$sig(\psi_{i,j}^t) = 1/(1 + \exp(-\psi_{i,j}^t)), \tag{8}$$

$$x_{i,j}^t = \begin{cases} 1 & \text{if} \quad r \le sig(\psi_{i,j}^t) \\ 0 & \text{if} \quad r > sig(\psi_{i,j}^t), \end{cases} \tag{9}$$

where $sig(\psi_{i,j}^t)$ is a sigmoid function and r is a uniform random number between [0, 1].

If $|V| = 6$, then the solution representation of a particle X_i is as follows.



k-node set (G_k)

Dimension	1	2	3	4	5	6
Position	0	1	0	1	0	1

Figure 1. Definition of particle.

Assuming that the fitness function $f(\cdot)$ is to be maximized, the fitness value of particle position can be evaluated by the weight of k-node set as follows.

$$f(position) = f(G_k) = w(G_k) - \nabla(G_k). \tag{10}$$

The particle best position of each particle is updated using the following equation.

$$pbest_i^t = \begin{cases} pbest_i^{t-1} & \text{if} \quad f(X_i^t) \le f(pbest_i^{t-1}) \\ X_i^t & \text{if} \quad f(X_i^t) > f(pbest_i^{t-1}). \end{cases} \tag{11}$$

Finally, the global best position found so far in the swarm population is obtained for $1 \le i \le ps$ as

$$gbest^t = \begin{cases} pbest_i^t & \text{where arg max } f(pbest_i^t) \\ & \text{if } \max f(pbest_i^t) > f(gbest^{t-1}) \\ gbest^{t-1} & \text{otherwise.} \end{cases} \tag{12}$$

3.2. The proposed algorithm

The particle swarm optimization algorithm to maximize k-node set reliability in a DS under capacity constraint presents as follows.

Algorithm PSODSR

Step 1. Read a DS topology $G = (V, E)$, $p_{i,j}$ of each link e_{ij}, $c(\nu_s)$ of each node ν_s and capacity constraint $C_{\min}$.

Step 2. Evaluate the weight $w(\nu_s)$ of each node ν_s using Eq. (1).

Step 3. Evaluate the weight $w(e_{s,d})$ of each link $e_{s,d}$ using Eq. (2).

Step 4. Evaluate the weight $w(\varepsilon_{s,d})$ of each pair node $\varepsilon_{s,d}$ using Eq. (3).

Step 5. Initial ps particles with random positions $X^0 = [X_1^0, X_2^0, \dots, X_{ps}^0]$. Generate velocities $\psi_{i,j}^0$, $i = 1, 2, \dots, ps$ and $j = 1, 2, \dots, n$, where $\psi_{i,j}^0$ is uniform random numbers between $[0, 1]$.

Step 6. For each particle

(a) Evaluate the weight $w(G_k)$ of selected set using Eq. (4).

(b) Evaluate the value $\nabla(G_k)$ of penalty function of selected set using Eq. (5).

(c) Evaluate fitness value using Eq. (10).

Step 7. Update individual and global best position according to Eq. (11) and Eq. (12), respectively.

Step 8. Update velocity: update the *i*th particle velocity using Eq. (6) restricted by maximum and minimum threshold $\Psi_{\max}$ and $-\Psi_{\max}$.

Step 9. Update position: update the *i*th particle position using Eq. (9).

Step 10. Update μ^t according to Eq. (7).

Step 11. Repeat Step 6 to 10 until a given maximum number of iterations $iter_{\max}$ is achieved.

Step 12. Compute $R(G_k)$ using SYREL, output the k-node set G_k and its reliability.

4. Comparison and Discussion

Table 1 presents the data on the results obtained using different methods for various DS topologies. In contrast to the EM and the KM, the number of reliability computations (NRC) grew rapidly when the DS topology size is increased.

Although capable of yielding the optimal solution, conventional techniques such as EM cannot effectively reduce the reliability count. An application occasionally requires an efficient algorithm to compute reliability owing to resource considerations. Under this circumstance, deriving the optimal reliability may not be feasible. Instead, an efficient algorithm yielding approximate reliability is preferred. Although the KM can reduce computational time in a moderate DS, the error from an exact solution is relatively high.

Table 1. Comparison with other methods

size		Exhaustive method			EM	KM		PSODSR	
N	e	Max_Rel	k-node set	NRC	NRC	NRC	absolute error	NRC	absolute error
6	9	0.9950069	1,3,5	64	20	9	0.0280687	1	0
7	11	0.9967785	1,2,3	128	35	11	0.0200188	1	0
8	11	0.9974378	4,6	256	44	13	0.0361958	1	0
10	13	0.9347952	1,7,8,9,10	1024	255	24	0.2155169	1	0
10	19	0.9995282	1,5,6	1024	150	17	0.0019527	1	0
11	17	0.9974023	1,10,11	2048	135	19	0.0120129	1	0
12	18	0.9858263	3,4,5,6	4096	538	30	0.0299250	1	0
12	21	0.9990777	1,3,5,6	4096	537	30	0.0056617	1	0.0006069
13	20	0.9978402	4,6	8192	246	45	0.0189070	1	0
19	31	0.9979870	6,8,9	524288	2369	37	0.0856661	1	0

The complexity of EM is $O(2^e \times 2^n)$ [2], where e denotes the number of edges and n represents the number of nodes. The complexity of the KM is $O(2^e \times n^2)$ [1]. In the PSODSR, in the worst case, the complexity of evaluating the weight of each node is $O(e)$ and each link is $O(e \times n)$, deriving an adequate k-node set is $O(ps \times n \times \text{max\_iter})$, and computing the reliability of the k-node set using SYREL is $O(m^2)$ [3]. Therefore, the complexity of the PSODSR is $\max(O(ps \times n \times \text{max\_iter}), O(m^2))$, where m represents the number of paths of a selected k-node set [3].

In our simulation case, the reliability count for the PSODSR is exactly one. The exact solution can be obtained high hit rate, in which the average error from exact solution is very slight. In a few cases, an adequate node which has arrived for selected node set through many paths and the length of a great number of those paths exceeds two, the node may be lost when using our formula for computing link's weight. Notably, the PSODSR cannot obtain the exact solution.

5. Conclusions

DS provides a cost-effective means of enhancing a computer system's performance in areas such as throughput, fault-tolerance, and reliability optimization. Consequently, the reliability optimization of a DS has become a critical issue. When some data files are allocated into DS, a specified set, K, of nodes in a DS must be selected to allocate the data files such that k-node set reliability is adequate under constraints.

We presented a PSODSR to obtain a k-node set with sub-optimal reliability. The PSODSR is based on not only a simple method to compute each node's weight and each link's weight, but also an efficient and effective objective function to evaluate the weight of node sets. The reliability computation in our algorithm is only exactly one.

The algorithm is compared with the EM and the KM for various topologies. According to that comparison, the PSODSR is more efficient in terms of execution time for a large DS. When the proposed method fails to provide an exact solution, the error from the exact solution is only slight.

References

[1] R. S. Chen, D. J. Chen and Y. S. Yeh, A new heuristic approach for reliability optimization of distributed computing systems subject to capacity constraints, Journal of Computers Mathematics with Application 29 (1995), 37-47.

[2] R. S. Chen, D. J. Chen and Y. S. Yeh, Reliability optimization of distributed computing systems subject to capacity constraint, Journal of Computers Mathematics with Application 29 (1995), 93-99.

[3] S. Hariri and C. S. Raghavendra, SYREL: a symbolic reliability algorithm based on path and cuset methods, IEEE Trans. Comput. C-36 (1987), 1224-1232.

[4] J. Kennedy and R. C. Eberhart, Particle swarm optimization, Proc. IEEE International Conference on Neural Networks, Piscataway, NJ, 1995, pp. 1942-1948.

[5] J. Kennedy and R. C. Eberhart, A discrete binary version of the particle swarm algorithm, Proc. Conf. Systems, Man, Cybernetics, Piscataway, NJ, 1997, pp. 4104-4108.

[6] Sangwook Lee, Haeaun Park and Moongu Jeon, Binary particle swarm optimization with bit change mutation, IEICE Transactions on Fundamentals E90-a (2007), 2253-2256.

[7] Ching-Jong Liao, Chao-Tang Tseng and Pin Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, Comput. Oper. Res. 34 (2007), 3099-3111.

[8] Rui Mendes, James Kennedy and Jose Neves, The fully informed particle swarm > simpler, maybe better, IEEE Transactions of Evolutionary Computation 1 (2005).

[9] Q. K. Pan, M. F. Tasgetiren and Y. C. Lian, A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, Comput. Oper. Res. 35 (2008), 2807-2839.

[10] A. Satyanarayna and J. N. Hagstrom, New algorithm for reliability analysis of multiterminal networks, IEEE Transactions on Reliability R-30 (1981), 325-333.

[11] S. N. Sivanandam, P. Visalakshi and A. Bhuvaneswari, Multiprocessor scheduling using hybrid particle swarm optimization with dynamically varying inertia, International Journal of Computer Science and Applications 4 (2007), 95-106.

[12] J. A. Stankovic, A perspective on distributed computer systems, IEEE Trans. Comput. 33 (1984), 1102-1115.

[13] R. S. Wilkov, Analysis and design of reliable computer networks, IEEE Transactions on Communications COM-20 (1970), 660-678.